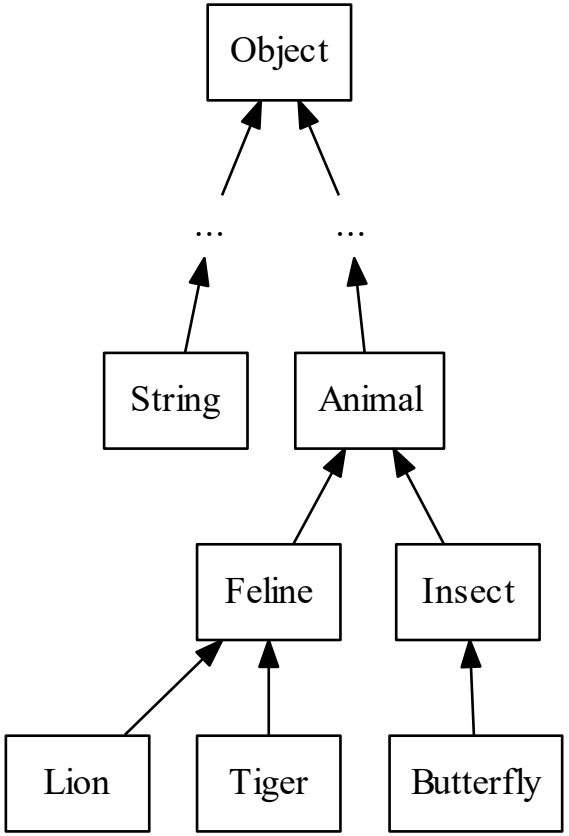
Java: Generics - Wildcard

IF2210 – Semester II 2020/2021

by: SAR

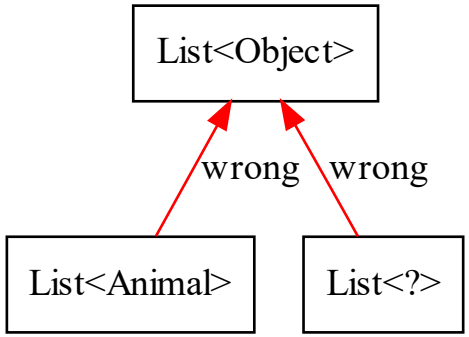
Wildcard

- Ingat kembali *subtyping*/relasi *is-a*.
- `Object` adalah leluhur semua kelas di Java.



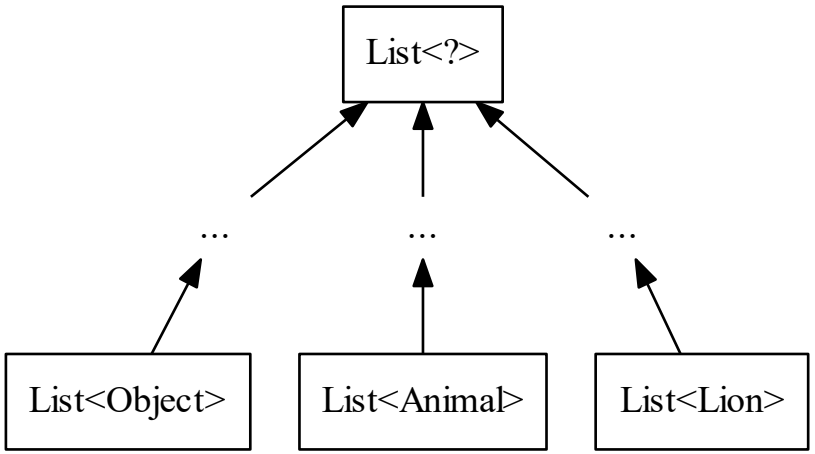
Wildcard

- Tapi untuk generik, e.g. `List<Object>` bukan leluhur dari semua *list*.



Wildcard

- Leluhur dari semua *list* adalah `List<?>`, dibaca sebagai *"list of unknown"*.
- Disebut juga *wildcard type*.



```java
void printListOfObjects(List<Object> l) {
    for (Object e: l) System.out.println(e);
}

void printListOfUnknowns(List<?> l) {
    for (Object e: l) System.out.println(e);
}

public static void main(String[] args) {
    List<Object> lo = new ArrayList<String>(); // compile error
    List<Object> lo = new ArrayList<Object>(); // OK
    List<?> lu = new ArrayList<String>();      // OK
    List<String> ls = new ArrayList<String>(); // OK

    printListOfObjects(lo); // OK
    printListOfObjects(lu); // compile error
    printListOfObjects(ls); // compile error

    printListOfUnknowns(lo); // OK
    printListOfUnknowns(lu); // OK
    printListOfUnknowns(ls); // OK

    // bersambung ...
```

```
    // ... sambungan

    lo.add("something");  // OK
    lu.add("something");  // compile error, String "is not an" unknown
    lu.add(new Object()); // compile error, Object "is not an" unknown
    ls.add("something");  // OK

    Object o = lo.get(0); // OK, get() returns Object
    Object u = lu.get(0); // OK, get() returns unknown,
                          // unknown "is an" Object
    Object s = ls.get(0); // OK
}
```

Bounded & unbounded wildcard

- Penggunaan *wildcard* pada contoh sebelumnya merupakan *unbounded wildcard* dengan sintaks `<?>`.
  - Yang diketahui hanya *unknown* is an `Object` (karena di Java apapun is an `Object`)
- Jenis *wildcard* lainnya adalah *bounded wildcard*:
  - *Upper bounded wildcard*: `<? extends Something>`
  - *Lower bounded wildcard*: `<? super Something>`

Upper bounded wildcard

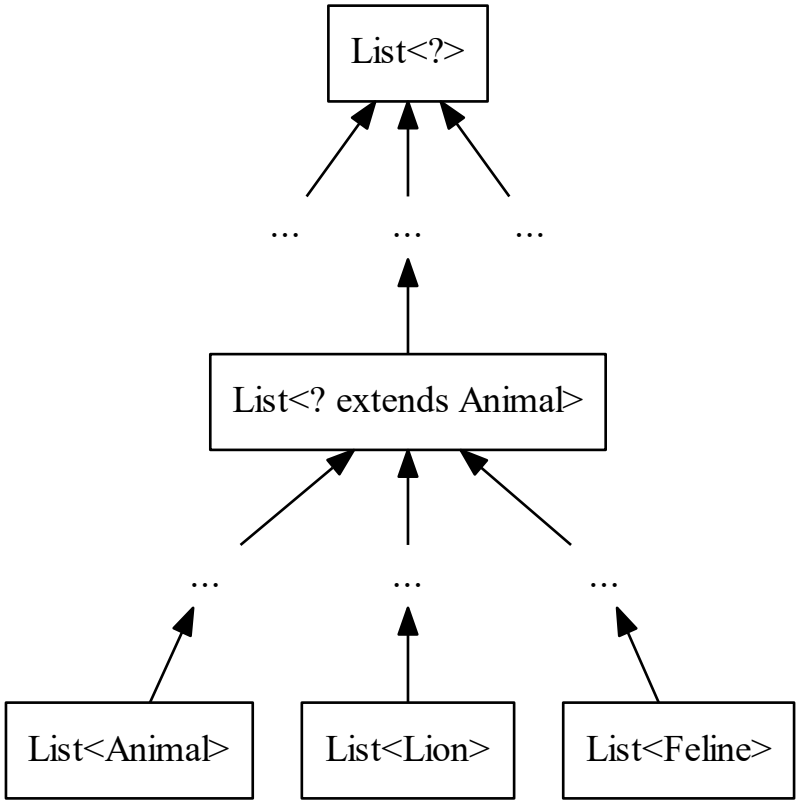
- Penggunaan *unbounded wildcard* menyebabkan variabel bertipe *wildcard* hanya dapat diperlakukan sebagai `Object`, seperti pada:

```
void printListOfUnknowns(List<?> l) {
    for (Object e: l)
        System.out.println(e);
}
```

- Pada contoh berikut, *upper bounded wildcard* membatasi `l` menjadi "list of sesuatu, di mana sesuatu adalah `Animal` atau turunannya":

```
void feedAnimals(List<? extends Animal> l) {
    for (Animal a: l)
        a.feedMe();
}
```

Ilustrasi upper bounded wildcard



```
void feedAnimals(List<? extends Animal> l) {
    for (Animal a: l)
        a.feedMe();
}

public static void main(String[] args) {

    List<? extends Animal> la = new ArrayList<Animal>();
    List<Lion> ll = new ArrayList<Lion>();
    List<Butterfly> lb = new ArrayList<Butterfly>();

    la = ll;
    feedAnimals(la);      // OK
    feedAnimals(ll);      // OK
    feedAnimals(lb);      // OK

    la.add(new Lion());   // still compile error, Lion is a known
                          // subclass of Animal

    Animal a = la.get(0); // OK, returns unknown subclass of Animal
}
```

Lower bounded wildcard

- Penggunaan *unbounded* ataupun *upper bounded wildcard* menyebabkan variabel tidak bisa digunakan untuk *update*, seperti contoh sebelumnya:
  - `la.get(0)` bisa, tetapi
  - `la.add(new Lion())` tidak bisa.
- Pada contoh berikut, *lower bounded wildcard* membatasi `l` menjadi "list of sesuatu, di mana sesuatu adalah `Feline` atau leluhurnya":

```
List<? super Feline> lf = new ArrayList<Feline>();
lf.add(new Lion());
```

```
public static void main(String[] args) {

    List<? super Feline> lf;
    lf = new ArrayList<Animal>(); // OK
    lf = new ArrayList<Feline>(); // OK
    lf = new ArrayList<Lion>();   // compile error

    Animal a = ...;
    Feline f = ...;
    Lion l = ...;
    lf.add(a); // compile error
    lf.add(f); // OK
    lf.add(l); // OK

    a = lf.get(0);         // compile error
    f = lf.get(0);         // compile error
    l = lf.get(0);         // compile error
    Object o = lf.get(0); // OK
}
```

Kesimpulan

- Dalam penggunaan *wildcard* di generik:
  - Jika variabel hanya dibaca, gunakan *upper bound*.
  - Jika variabel hanya untuk modifikasi, gunakan *lower bound*.
  - Jika variabel dibaca dan untuk modifikasi, jangan gunakan *wildcard*.